v7.n11.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

DESENVOLVIMENTO DE UMA PLATAFORMA WEB CENTRALIZADA PARA GESTÃO FINANCEIRA: UMA ALTERNATIVA INCLUSIVA E ESTRUTURADA ÀS PLANILHAS ELETRÔNICAS

DEVELOPMENT OF A CENTRALIZED WEB PLATFORM FOR FINANCIAL MANAGEMENT: AN INCLUSIVE AND STRUCTURED ALTERNATIVE TO SPREADSHEETS

DESARROLLO DE UNA PLATAFORMA WEB CENTRALIZADA PARA LA GESTIÓN FINANCIERA: UNA ALTERNATIVA INCLUSIVA Y ESTRUCTURADA A LAS HOJAS DE CÁLCULO

Matheus Figueira Merlos[1], Lucas Diniz Dallilo[2], Fabiana Florian[3]

e7119528
https://doi.org/10.47820/recima21.v7i11.9528

PUBLICADO: 11/2026

RESUMO

O elevado número de inadimplentes no Brasil, que atingiu 81,3 milhões em janeiro de 2026, evidencia a necessidade de ferramentas de controle financeiro pessoal mais confiáveis do que as planilhas eletrônicas, sujeitas a erros de cálculo e com validação de dados limitada. Este trabalho teve como objetivo desenvolver uma plataforma *web* centralizada de gestão financeira pessoal, com validação automatizada de dados, precisão aritmética e recursos de acessibilidade, como alternativa às planilhas. Trata-se de pesquisa aplicada, de natureza descritiva e explicativa e abordagem qualitativa. A plataforma foi construída com *back-end* em Java e Spring Boot, banco de dados Microsoft SQL Server e interface em Angular, e validada por testes manuais das requisições à API com a ferramenta Postman e por comparação qualitativa com o método em planilhas. Os testes confirmaram as respostas esperadas para lançamentos válidos, dados inválidos e acessos sem autenticação, e o tipo BigDecimal produziu saldo exato de R$ 0,20 em um cálculo no qual o tipo double resultaria em 0,19999999999999998. Os resultados indicam que a plataforma reúne precisão aritmética, segurança da informação e recursos de acessibilidade em um único ambiente, constituindo alternativa estruturada às planilhas, ainda que a avaliação tenha se limitado à observação do autor.

PALAVRAS-CHAVE: Gestão financeira pessoal. Engenharia de *software*. Acessibilidade digital. Integridade de dados. Aplicação *web*.

ABSTRACT

The high number of defaulters in Brazil, which reached 81.3 million in January 2026, highlights the need for personal financial management tools more reliable than electronic spreadsheets, which are prone to calculation errors and offer limited data validation. This study aimed to develop a centralized web platform for personal financial management, with automated data validation, arithmetic precision, and accessibility features, as an alternative to spreadsheets. This is applied research, descriptive and explanatory in nature, with a qualitative approach.

[1] Graduando em Engenharia de Computação pela Universidade de Araraquara, com atuação em testes de software e banco de dados.
[2] Mestre em Ciência da Computação pela Universidade Federal de São Carlos, doutorando pela Universidade de São Paulo e docente da Universidade de Araraquara (UNIARA), na área de Engenharia de Software.
[3] Doutora em Alimentos e Nutrição pela Universidade Estadual Paulista, docente da Universidade de Araraquara (UNIARA), com atuação em Economia e Metodologia Científica.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

The platform was built with a Java and Spring Boot back-end, a Microsoft SQL Server database, and an Angular interface, and was validated through manual tests of API requests using the Postman tool and through a qualitative comparison with the spreadsheet-based method. The tests confirmed the expected responses for valid entries, invalid data, and unauthenticated access, and the BigDecimal type produced an exact balance of R$ 0.20 in a calculation in which the double type would yield 0.19999999999999998. The results indicate that the platform combines arithmetic precision, information security, and accessibility features in a single environment, constituting a structured alternative to spreadsheets, although the evaluation was limited to the author's observation.

KEYWORDS: *Personal financial management. Software engineering. Digital accessibility. Data integrity. Web application.*

RESUMEN
El elevado número de morosos en Brasil, que alcanzó 81,3 millones en enero de 2026, evidencia la necesidad de herramientas de control financiero personal más confiables que las hojas de cálculo electrónicas, sujetas a errores de cálculo y con validación de datos limitada. Este trabajo tuvo como objetivo desarrollar una plataforma web centralizada de gestión financiera personal, con validación automatizada de datos, precisión aritmética y recursos de accesibilidad, como alternativa a las hojas de cálculo. Se trata de una investigación aplicada, de naturaleza descriptiva y explicativa y enfoque cualitativo. La plataforma se construyó con back-end en Java y Spring Boot, base de datos Microsoft SQL Server e interfaz en Angular, y se validó mediante pruebas manuales de las solicitudes a la API con la herramienta Postman y mediante una comparación cualitativa con el método en hojas de cálculo. Las pruebas confirmaron las respuestas esperadas para registros válidos, datos inválidos y accesos sin autenticación, y el tipo BigDecimal produjo un saldo exacto de R$ 0,20 en un cálculo en el que el tipo double resultaría en 0,19999999999999998. Los resultados indican que la plataforma reúne precisión aritmética, seguridad de la información y recursos de accesibilidad en un único entorno, y constituye una alternativa estructurada a las hojas de cálculo, aunque la evaluación se haya limitado a la observación del autor.

PALABRAS CLAVE: *Gestión financiera personal. Ingeniería de software. Accesibilidad digital. Integridad de datos. Aplicación web.*

INTRODUÇÃO

No cenário brasileiro recente, a gestão financeira se tornou determinante para a estabilidade econômica dos indivíduos. Segundo o Mapa da Inadimplência da Serasa Experian (2026), janeiro de 2026 registrou 81,3 milhões de inadimplentes, o maior número de toda a série histórica, após 13 meses consecutivos de alta. Esses dados reforçam a necessidade de ferramentas que auxiliem a população na gestão correta de seus gastos, com foco em conquistar a estabilidade financeira.

O acompanhamento do orçamento pessoal é importante para manter a estabilidade financeira no longo prazo. Mesmo que grande parte da sociedade utilize controles manuais baseados em planilhas eletrônicas, elas oferecem validação de dados limitada, que depende de

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

configuração manual, e recursos de acessibilidade restritos. Outro ponto importante é que as planilhas não protegem de forma estrutural contra erros de digitação ou perda de referências de cálculo.

Além da validação limitada, o grande problema das planilhas é que elas escondem erros de cálculo que normalmente não são vistos. Confiar nessas ferramentas pode custar caro, pois um estudo identificou falhas na lógica matemática de planilhas eletrônicas, como a ordem de resolução de potências, que geram resultados incorretos e só são percebidas por usuários com conhecimento matemático (MEDEIROS; GONÇALVES, 2020). Para quem busca organizar as finanças pessoais e não possui domínio avançado de matemática, esse modelo representa um risco, visto que um erro de preenchimento pode comprometer todo o planejamento. Diante desses problemas, a saber, a validação automática limitada dos dados, a ocorrência de erros de cálculo não percebidos pelo usuário, a vulnerabilidade a erros de digitação e a limitação dos recursos de acessibilidade, questiona-se: de que forma uma plataforma *web* centralizada, com validação automatizada de dados e recursos de acessibilidade, pode superar as limitações das planilhas eletrônicas no controle financeiro pessoal?

A pesquisa justifica-se pela relevância social da organização financeira doméstica, uma vez que maior educação financeira está associada a menor propensão ao endividamento (SILVA; BUBECK; RODRIGUES JÚNIOR, 2025). Embora existam aplicativos de controle financeiro, a acessibilidade ainda é um desafio nesse tipo de ferramenta (MELO *et al.*, 2024), e este trabalho propõe reunir, em uma mesma plataforma, validação automatizada, precisão aritmética, recursos de acessibilidade e importação dos dados que o usuário já mantém em planilhas. No âmbito acadêmico, o trabalho demonstra a aplicação de conceitos de Engenharia de Software na resolução de um problema cotidiano.

O objetivo geral deste trabalho é desenvolver uma plataforma *web* centralizada de gestão financeira pessoal, estruturada e acessível, como alternativa às planilhas eletrônicas, inclusive para quem deseja migrar os dados que já possui. Para alcançá-lo, estabelecem-se os seguintes objetivos específicos: (i) implementar a arquitetura da plataforma, com validação de dados e precisão aritmética nos cálculos financeiros; (ii) incorporar mecanismos de segurança da informação, recursos de acessibilidade e a importação de dados de planilhas em formato CSV; e (iii) comparar qualitativamente a plataforma com o método tradicional em planilhas, quanto à facilidade de uso, precisão, segurança e acessibilidade.

Quanto à metodologia, trata-se de uma pesquisa aplicada, de natureza descritiva e explicativa, com abordagem qualitativa. Foi realizada pesquisa bibliográfica em livros, publicações acadêmicas e documentos institucionais, que fundamentou o desenvolvimento da

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

plataforma. A validação ocorreu por testes das requisições HTTP à API e pela conferência dos cálculos com valores calculados manualmente, seguidos de uma comparação qualitativa com o método em planilhas. Os procedimentos são detalhados na seção 2.

Além desta introdução, o artigo está organizado em cinco seções. A seção 1 apresenta o referencial teórico; a seção 2 descreve a metodologia; a seção 3 apresenta o desenvolvimento da plataforma; a seção 4 apresenta e discute os resultados; e a seção 5 traz as considerações finais.

1. REFERENCIAL TEÓRICO

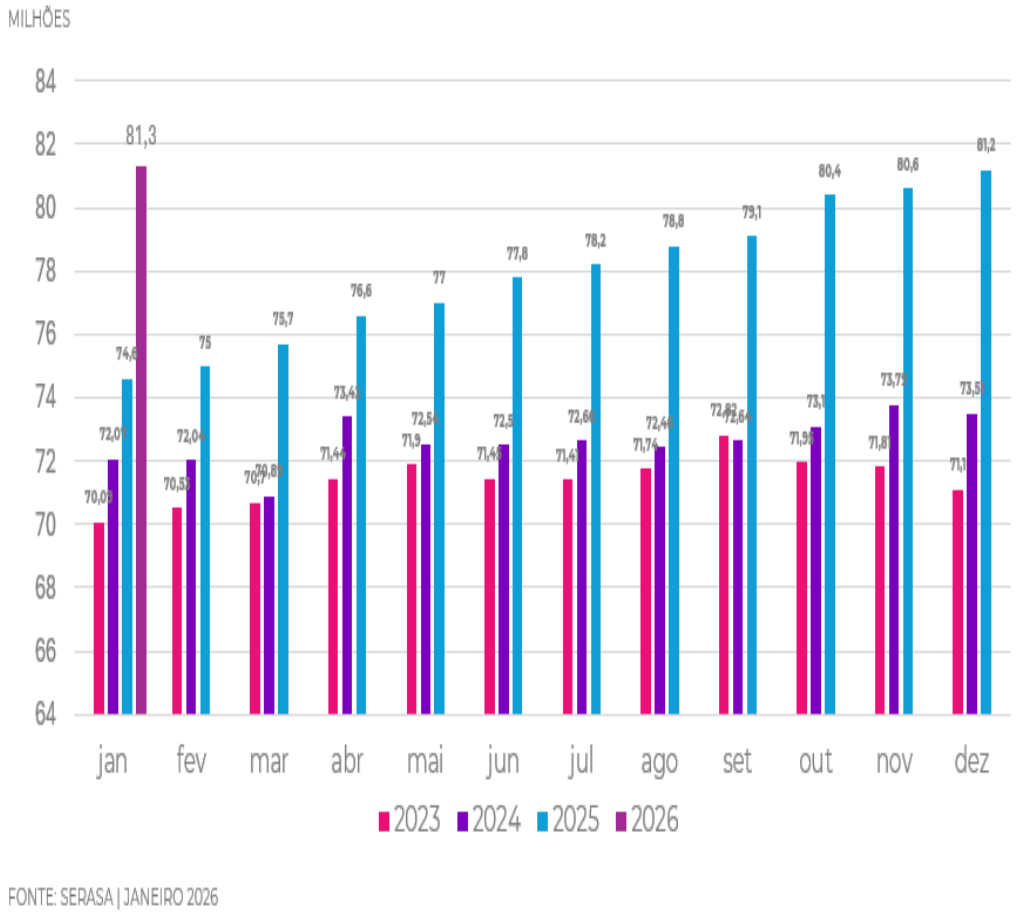
Nesta seção são abordadas as limitações das planilhas eletrônicas frente a um sistema estruturado, os princípios da arquitetura *web* cliente-servidor e das APIs RESTful, os fundamentos da linguagem Java e do *framework* Spring Boot utilizados no *back-end*, o papel da segurança da informação com autenticação via *tokens* JWT, o banco de dados relacional Microsoft SQL Server, e, por fim, o ecossistema Angular com TypeScript adotado na construção da interface intuitiva para o usuário.

1.1. O cenário da inadimplência e as limitações das planilhas no controle financeiro

O Mapa da Inadimplência da Serasa Experian (2026) mostra que o país atingiu o maior número de inadimplentes de toda a sua série histórica, conforme a Figura 1, que apresenta a evolução entre 2023 e janeiro de 2026, o que sugere que o problema não é pontual, mas estrutural. Esse cenário pode estar relacionado à perda do poder de compra causada pela inflação e pelos juros altos, que tornam o crédito rotativo mais caro, à liberação facilitada de empréstimos e à baixa instrução financeira da população, que leva muitos cidadãos a recorrer ao crédito para cobrir o déficit no orçamento mensal. Nesse sentido, em pesquisa com 318 estudantes de pós-graduação, Silva, Bubeck e Rodrigues Júnior (2025) verificaram que uma maior educação financeira está associada a uma menor propensão ao endividamento. Diante disso, ferramentas de gestão financeira pessoal tornam-se essenciais para que o cidadão consiga controlar o orçamento de forma contínua e prevenir o endividamento.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 1. Evolução do número de inadimplentes no Brasil (2023 a jan. 2026)



Fonte: Serasa Experian (2026).

O uso de planilhas eletrônicas para o controle financeiro apresenta riscos à integridade da informação, porque essas ferramentas possuem mecanismos de validação opcionais e limitados dos dados inseridos. Em contrapartida, sistemas estruturados garantem a confiabilidade ao aplicar restrições diretas no banco de dados. Sobre a importância desse controle em sistemas gerenciadores, Kroenke et al. (2018, p. 15, tradução nossa) explicam que "com a maioria dos SGBD, é possível declarar regras sobre os valores dos dados e fazer com que o SGBD as aplique. [...] O SGBD então aplica essas regras, chamadas de restrições de integridade referencial".

Nesse contexto financeiro, a tipagem pouco rigorosa e as validações automáticas limitadas das planilhas, diferentemente do que ocorre nos bancos de dados, permitem a entrada

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

de dados inconsistentes, o que pode resultar em erros acumulados, comprometendo a confiabilidade das informações.

1.2. Pilares tecnológicos e ferramentas de implementação

Sistemas *web* são compostos, em geral, por uma camada de apresentação, executada no navegador do usuário (*front-end*), e por uma camada de servidor, responsável pelo processamento das regras de negócio e pelo acesso aos dados (*back-end*). As subseções a seguir apresentam os fundamentos de cada camada e as tecnologias associadas a elas.

1.2.1. Interface gráfica e camada de apresentação (*front-end*)

A programação no lado do cliente permite a visualização do sistema no navegador do usuário. O JavaScript, por exemplo, é uma linguagem fundamental na *Web*, amplamente utilizada para adicionar funcionalidades dinâmicas.

Nesse sentido, a utilização do ecossistema Angular, que integra nativamente as linguagens de estruturação e estilo (HTML5 e CSS3) ao TypeScript (uma evolução do JavaScript), permite a construção da interface gráfica (*front-end*). Essa abordagem substitui a visualização estática das planilhas eletrônicas por um ambiente interativo, no qual o usuário consegue gerenciar suas finanças de maneira simplificada.

O ecossistema Angular utiliza o padrão de arquitetura SPA (*Single Page Application*), em que a aplicação é carregada uma única vez pelo navegador e só as telas mudam dinamicamente sem recarregar a página. Segundo a MDN Web Docs (2025), esse modelo permite ao usuário navegar sem carregar páginas inteiras do servidor, o que pode resultar em ganhos de desempenho e em uma experiência mais dinâmica. Aplicado ao controle financeiro, esse comportamento favorece a navegação do usuário entre relatórios, lançamentos e resumos de forma rápida, sem travamentos visuais.

1.2.2. Camada de servidor e regras de negócio (*back-end*)

A linguagem Java é robusta e possui em sua arquitetura a orientação a objetos, que facilita a organização de sistemas complexos. Conforme explicam Deitel e Deitel (2017, p. 14), a utilização da *Java Virtual Machine* (JVM) garante a portabilidade do sistema, permitindo que a aplicação possa ser executada em diferentes ambientes. Junto com isso, a tipagem forte da linguagem determina uma confiabilidade no tratamento de dados sensíveis. O uso do *framework* Spring Boot justifica-se por oferecer configuração padronizada para a criação de APIs REST. De acordo com a documentação oficial do projeto (SPRING, c2026), o Spring Boot configura automaticamente o Spring e bibliotecas de terceiros sempre que possível, e a maioria das

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

aplicações precisa de configuração mínima, o que permite concentrar o desenvolvimento na lógica de negócio. Assim, o servidor atua como um controlador que garante a segurança das transações financeiras antes de qualquer registro no banco de dados.

A segurança da plataforma foi desenvolvida em duas camadas que se completam, com proteção da senha e controle de sessão. Para controlar a senha, o sistema usa o Spring Security com o algoritmo BCrypt, que transforma a senha do usuário em um *hash* irreversível antes de salvar no banco de dados, para garantir que, mesmo se o banco for acessado por algum invasor, a senha continue protegida. Já para o controle de acesso, foi adotado o padrão JWT (*JSON Web Token*), definido pela RFC 7519 (JONES; BRADLEY; SAKIMURA, 2015) como um meio compacto de representar declarações transferidas entre duas partes, que podem ser assinadas digitalmente. Nessa abordagem, o servidor gera um *token* com prazo de expiração, que o cliente reenvia a cada requisição, e como as informações de identificação viajam no próprio *token*, o servidor não precisa armazenar sessões. Em contrapartida, como o *token* é autossuficiente e não pode ser facilmente revogado antes de expirar, o prazo de expiração definido torna-se um parâmetro importante em uma plataforma que lida com dados financeiros.

1.2.3. Arquitetura de comunicação e camada de persistência

A integração entre as camadas de interface e o servidor ocorre por meio do padrão REST (*Representational State Transfer*). Essa arquitetura estabelece uma separação entre as responsabilidades do cliente e do servidor, utilizando o protocolo HTTP para a troca de dados em formato JSON. Essa independência torna a manutenção do sistema mais simples, uma vez que o *back-end* se concentra na lógica de negócio e na persistência, enquanto o *front-end* gerencia a experiência do usuário.

Para a comunicação entre a aplicação Java e o banco de dados SQL Server, o sistema utiliza o Spring Data JPA, que implementa o conceito de ORM (*Object-Relational Mapping*). Por meio do *framework* Hibernate, esse mecanismo realiza o mapeamento automático entre as classes Java e as tabelas do banco de dados relacional, eliminando a necessidade de escrita manual de instruções SQL para as operações de consulta, inserção, atualização e exclusão de dados. Essa abstração reduz a complexidade do acesso a dados e minimiza a ocorrência de erros na manipulação das informações persistidas, contribuindo diretamente para a confiabilidade dos registros financeiros da plataforma.

1.2.4. Padrão arquitetural *Model-View-Controller* (MVC)

A arquitetura do sistema baseia-se no padrão MVC (*Model-View-Controller*), que, conforme definido por Sommerville (2011), organiza a aplicação em três componentes

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

interconectados para separar a representação interna da informação da forma como ela é apresentada ao usuário. Nesta aplicação, o *Model* e o *Controller* ficam no *back-end*, e o papel de *View* é exercido pela interface Angular, que consome a API REST como cliente desacoplado.

1.3. Persistência de dados e o banco de dados relacional

Para o armazenamento persistente de informações financeiras, os Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR), como o SQL Server, oferecem mecanismos de integridade e de controle de transações (ELMASRI; NAVATHE, 2019). O modelo relacional representa os dados e seus relacionamentos em tabelas, o que permite reduzir redundâncias por meio da normalização.

Para a comunicação com essas tabelas, a linguagem SQL (*Structured Query Language*) atua como meio de interação, abrangendo comandos para definir, manipular e controlar o acesso de segurança aos dados. Vale destacar também que os sistemas de bancos de dados relacionais garantem que as transações ocorram sob as propriedades ACID, isto é, atomicidade, consistência, isolamento e durabilidade (ELMASRI; NAVATHE, 2019). Isso significa que uma transferência ou o registro de um gasto só é concluído se todas as etapas da operação tiverem sucesso, evitando que falhas no sistema resultem em saldos incorretos, por exemplo.

1.4. Acessibilidade e inclusão digital em sistemas *web*

A transição das planilhas para uma plataforma *web* também precisa considerar a acessibilidade. De acordo com a linha do tempo "Do eletrônico ao digital", do portal Governo Digital (BRASIL, 2019), a transformação digital dos serviços públicos buscou a desburocratização, a simplificação de processos e a prestação de serviços sem a necessidade da presença física, o que reforça a importância de plataformas centradas no usuário.

Segundo o Modelo de Acessibilidade em Governo Eletrônico (eMAG), o objetivo da acessibilidade *web* é tornar o conteúdo acessível a todas as pessoas, inclusive às pessoas com deficiência (BRASIL, 2014).

Além das planilhas, há trabalhos que propõem aplicativos para o controle financeiro, como o de Andrade *et al.* (2024), que desenvolveram um aplicativo para o controle de despesas de estudantes universitários. Quanto à acessibilidade, Melo *et al.* (2024) inspecionaram cinco aplicativos bancários brasileiros considerando usuários com baixo letramento e propuseram melhorias aos desenvolvedores, o que indica que a acessibilidade ainda é um desafio também nos aplicativos financeiros.

Esses estudos indicam que a acessibilidade deve ser considerada desde a concepção de aplicações financeiras, e não como recurso complementar.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

1.5. Importação e migração de dados

A importação de arquivos CSV foi incluída para que usuários de planilhas não precisem digitar novamente o histórico de lançamentos, permitindo a transição para o sistema sem perda dos registros já existentes. A migração de dados é uma tarefa crítica, que exige planejamento para reduzir riscos e preservar a integridade e a segurança das informações durante a transferência entre sistemas (MORAES; MISAEL; BRAYNER, 2026).

2. METODOLOGIA

2.1. Caracterização da pesquisa

Trata-se de uma pesquisa aplicada, por buscar a solução de um problema prático, o controle financeiro pessoal em planilhas eletrônicas, por meio do desenvolvimento de uma plataforma *web*. Quanto aos objetivos, é descritiva, ao descrever a arquitetura e o funcionamento da plataforma, e explicativa, ao analisar as causas dos problemas identificados durante o desenvolvimento, como os erros de arredondamento. A abordagem é qualitativa. A fundamentação teórica foi construída por pesquisa bibliográfica em livros, artigos científicos, documentação técnica e documentos institucionais.

O estudo foi conduzido em quatro fases: (i) pesquisa bibliográfica; (ii) levantamento dos requisitos funcionais e não funcionais; (iii) desenvolvimento da plataforma; e (iv) validação, por testes da API e por comparação com o método em planilhas.

Ferramentas de inteligência artificial generativa foram utilizadas como apoio na elaboração do trabalho, conforme detalhado na declaração ao final do artigo. A plataforma foi executada em ambiente local, em contêineres Docker, sem hospedagem em serviços externos.

2.2. Etapas do desenvolvimento

O desenvolvimento seguiu cinco etapas, da camada de dados até a interface com o usuário: (1) modelagem das entidades e do banco de dados; (2) camada de persistência, com o padrão *Repository*; (3) camada de serviço, com as regras de negócio e a validação dos dados; (4) *endpoints* REST, com objetos de transferência de dados (DTOs) e configuração de CORS; e (5) interface *web* em Angular. As etapas são detalhadas na seção 3 e representadas na Figura 3.

2.3. Tecnologias e ambiente de execução

As tecnologias utilizadas e suas funções na plataforma são apresentadas no Quadro 1.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Quadro 1. Tecnologias utilizadas no desenvolvimento da plataforma

Camada	Tecnologia	Versão	Função
Back-end	Java (Eclipse Temurin)	17	Linguagem do servidor
Back-end	Spring Boot	3.2.3	Criação da API REST
Back-end	Spring Security	6.2.2	Autenticação e autorização
Back-end	jjwt (JWT com HMAC-SHA256)	0.11.5	Geração e validação dos *tokens* de acesso
Back-end	Spring Data JPA e Hibernate	3.2.3 e 6.4.4	Mapeamento objeto-relacional
Back-end	Bean Validation	Incluída no Spring Boot	Validação declarativa dos dados de entrada
Back-end	OpenPDF	1.3.43	Geração do relatório em PDF
Banco de dados	Microsoft SQL Server	2022 (16.0.4305.5)	Armazenamento persistente
Front-end	Angular e TypeScript	20.3 e 5.9.3	Interface *web* (SPA)
Infraestrutura	Docker e Docker Compose	—	Execução em contêineres
Testes	Postman	—	Testes manuais das requisições à API

Fonte: Elaborado pelo autor (2026).

Os testes foram realizados em ambiente local, com a plataforma executada em contêineres Docker orquestrados pelo arquivo docker-compose.yml, que inicia quatro serviços: o banco de dados SQL Server 2022, um serviço que cria o banco da aplicação, a API Spring Boot e a interface Angular, servida pelo nginx. As versões ficam fixadas nos *Dockerfiles*, e a pilha completa é iniciada com um único comando, o que permite reproduzir o mesmo ambiente em qualquer máquina com Docker instalado.

2.4. Procedimentos de validação

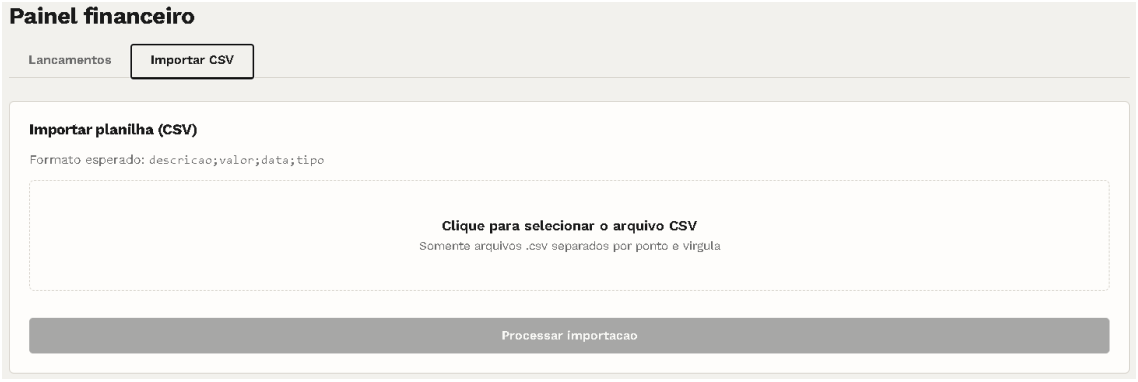
A validação técnica foi realizada por testes manuais das requisições HTTP à API, com a ferramenta Postman e a plataforma em execução. Foram executados os seis casos de teste apresentados na Tabela 1, que cobrem lançamento válido, valor negativo, campo obrigatório vazio, acesso sem autenticação, cálculo de saldo e importação de planilha. Em cada caso, conferiram-se o código de status HTTP e o corpo da resposta em JSON; nas requisições protegidas, o *token* JWT obtido no *login* foi enviado no cabeçalho Authorization. A precisão

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

aritmética foi verificada pela comparação entre o saldo calculado pela API e o valor calculado manualmente.

A plataforma lê arquivos .csv separados por vírgula, ponto e vírgula ou tabulação, como os exportados pelo Excel e pelo Google Sheets. No escopo da aplicação, cada registro textual corresponde a uma movimentação de receita ou despesa. Antes que ocorra a gravação definitiva na base de dados, rotinas interceptam possíveis inconsistências, e as linhas recusadas são informadas ao usuário com o número da linha e o motivo, assegurando a validade das informações financeiras manipuladas pelo usuário, conforme ilustrado na Figura 2. A importação foi testada com um arquivo contendo uma linha de valor inválido, verificando se essa linha era recusada e as demais, gravadas.

Figura 2. Módulo de importação de dados de planilhas legadas (CSV)



Fonte: Elaborado pelo autor (2026).

2.5. Procedimento de comparação com o método em planilhas

A comparação com o método tradicional foi realizada pelo autor, que executou as mesmas tarefas na plataforma e em três planilhas de controle financeiro elaboradas no Microsoft Excel: o cadastro de uma receita e de uma despesa e a consulta do saldo do mês. Na plataforma, também foi realizada a importação do histórico de lançamentos a partir de um arquivo CSV. As ferramentas foram comparadas qualitativamente quanto à facilidade de uso, à eficiência na execução das tarefas, à segurança dos dados e aos recursos de acessibilidade. Por ter sido conduzida apenas pelo autor, a comparação tem caráter exploratório, conforme indicado nas limitações (seção 5).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3. DESENVOLVIMENTO

Para o desenvolvimento do sistema são apresentadas as etapas, desde o processo de planejamento até o desenvolvimento das regras de negócios e códigos. Durante o desenvolvimento, houve dois problemas principais que precisaram ser corrigidos para que o sistema funcionasse da maneira correta. O primeiro ocorreu na importação de planilhas: valores como "350.50" eram convertidos para 35050, porque todos os pontos eram removidos, e tipos de movimentação inválidos eram convertidos silenciosamente em despesa. Por isso, a conversão e a validação passaram a ser feitas no *back-end*, linha a linha, antes da gravação, e as linhas inválidas passaram a ser recusadas e informadas ao usuário; o segundo foi que, durante os testes, os cálculos e saldos estavam sendo gerados incorretamente, devido a uma inconsistência na tipagem primitiva que causava erros de arredondamento sucessivos. A solução foi alterar a tipagem de dados flutuantes tanto no banco de dados quanto no código, garantindo que os cálculos passassem a ser realizados corretamente.

A construção da plataforma fundamenta-se no levantamento e na definição de requisitos técnicos. Os requisitos funcionais descrevem as ações que o *software* deve executar, como o cadastro de transações, a importação de dados de planilhas e a consulta de saldo, enquanto os requisitos não funcionais estabelecem os critérios de qualidade e restrições do sistema, com foco em precisão aritmética, usabilidade, acessibilidade e, principalmente, segurança da informação.

3.1. Etapas de desenvolvimento do sistema

O desenvolvimento foi organizado em cinco etapas (Figura 3), apresentadas na ordem em que foram realizadas, da camada de dados até a interface com o usuário.

Primeiramente, houve a modelagem e definição das entidades. A criação se iniciou pela classe Transacao e foram estabelecidos os dados fundamentais. O tipo BigDecimal foi adotado para os campos de valores, evitando os erros de arredondamento identificados nos testes iniciais.

Após isso, foi desenvolvida a camada de persistência dos dados com os modelos já definidos, em que o foco foi desenvolver a camada de acesso aos dados utilizando o padrão *Repository*. Essa camada abstrai as consultas ao banco de dados e permite que o sistema interaja com o banco de dados SQL Server aplicando as regras de integridade definidas no banco de dados.

Com uma camada de dados definida, a próxima etapa foi o desenvolvimento da camada de serviço. Foram implementadas as regras de negócio e a validação dos dados, que impedem a gravação de valores inconsistentes. A validação foi implementada de forma declarativa: o

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

objeto de entrada recebe anotações do Bean Validation (@NotBlank, @Positive, @Pattern), verificadas automaticamente pela anotação @Valid antes de a requisição chegar à camada de serviço. Na importação de planilhas, cada linha é validada individualmente, e as recusadas são informadas ao usuário com o número da linha e o motivo, sem impedir a gravação das linhas válidas.

Em seguida, ocorreu o desenvolvimento dos *endpoints* REST, que realizam a comunicação entre o *back-end* Java e o *front-end* Angular. Foi configurado o CORS no Spring Boot, liberando a origem da aplicação Angular (localhost:4200, em desenvolvimento) para que ela, hospedada em origem diferente, pudesse se comunicar com a API Java, viabilizando a troca de dados estruturados em JSON entre cliente e servidor, conforme ilustrado no fluxograma da Figura 3.

Para garantir a consistência das informações, adotou-se o uso de objetos de transferência de dados (DTOs) e validações estruturadas nas requisições, assegurando que o sistema processe apenas entradas íntegras e padronizadas. Essa padronização da API garantiu o desacoplamento das camadas e a eficiência no tráfego de dados entre cliente e servidor.

Por fim, foi desenvolvida a interface *web* em Angular, que consome os *endpoints* REST disponibilizados pela API e apresenta ao usuário o cadastro de movimentações, o histórico de transações e o painel financeiro, detalhados nas seções 3.4 e 3.5.

Figura 3. Fluxograma das etapas de desenvolvimento e arquitetura do sistema



Fonte: Elaborado pelo autor (2026).

3.2. Segurança da informação e autenticação

Para manter o controle de acesso seguro, foi adotado o *framework* Spring Security, responsável por estruturar uma camada de autenticação e autorização dos usuários. Essa ferramenta atua na proteção contra vulnerabilidades recorrentes da arquitetura *web*. Como a API

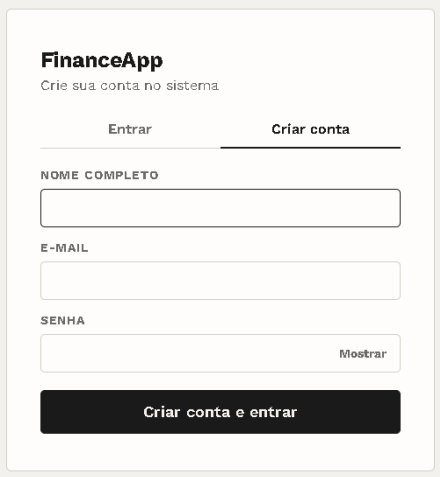
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

é *stateless* e a autenticação ocorre por *token* JWT enviado a cada requisição, e não por *cookies* de sessão, reduz-se a exposição a ataques do tipo *Cross-Site Request Forgery* (CSRF). Por esse motivo, a proteção contra CSRF do Spring Security foi desabilitada, já que o *token* não é enviado em *cookie* e o navegador não o anexa automaticamente a requisições de outros sites. O *token* é assinado com o algoritmo HMAC-SHA256 e tem validade de 24 horas.

Como medida preventiva, o sistema emprega o algoritmo BCrypt (baseado na cifra Blowfish). Diferentemente dos métodos tradicionais de criptografia bidirecional, o BCrypt gera uma representação irreversível da senha. Seu diferencial é o fator de custo (work factor), que insere um atraso computacional proposital em cada ciclo de geração, o que torna ataques de força bruta (*brute force*) significativamente mais difíceis. As diretrizes do NIST determinam que senhas sejam armazenadas com *salt* e *hash*, por meio de um esquema de *hash* de senhas, e recomendam que o fator de custo seja o mais alto possível sem prejudicar o desempenho do sistema (TEMOSHOK et al., 2025).

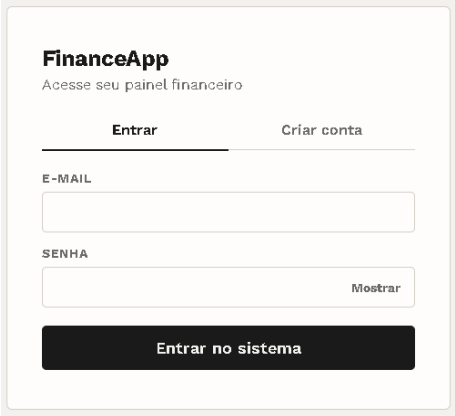
O sistema captura a senha inserida em texto plano no cadastro e a submete à função de *hash* antes de efetivar a persistência. Na autenticação (*login*), a senha informada é combinada ao *salt* (valor aleatório gerado pelo BCrypt e armazenado junto ao *hash*) e processada sob o mesmo fator de custo armazenado na base de dados. Por meio da validação matemática entre os códigos gerados, a identidade é confirmada e o acesso liberado sem que a credencial original seja armazenada, conforme ilustrado nas Figuras 4 e 5. Para proteger a senha durante o envio, a comunicação entre cliente e servidor deve ocorrer via HTTPS.

Figura 4. Cadastro de usuários



Fonte: Elaborado pelo autor (2026).

Figura 5. Login dos usuários



Fonte: Elaborado pelo autor (2026).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

A arquitetura *stateless* baseada em JWT, já adotada para o controle de acesso, também contribui para a segurança das comunicações entre *front-end* e *back-end*, reduzindo a dependência de sessões armazenadas no servidor.

Nessa lógica, a arquitetura escolhida mantém a integridade dos dados durante todo o ciclo da requisição. A implementação do Spring Security assegura que apenas usuários autenticados possuam permissão para realizar operações na camada de dados. Essa camada de autorização possui filtros de segurança que interceptam as chamadas da API, validando não apenas a identidade do usuário, mas também suas permissões de acesso aos recursos, protegendo o *back-end* de requisições não autorizadas. Além disso, a validação das entradas e o uso do JPA com consultas parametrizadas reduzem o risco de injeção de SQL. O *token* é armazenado no *localStorage* do navegador e enviado no cabeçalho Authorization a cada requisição, o que exige cuidado com ataques de XSS.

3.3. Tratamento de exceções e resiliência da API

Para assegurar a estabilidade da plataforma desenvolvida voltada ao controle financeiro, é necessário o gerenciamento adequado de falhas. Nesse tipo de sistema, exceções inesperadas, como, por exemplo, valores inconsistentes, ausência de parâmetros obrigatórios ou interrupções na comunicação com o banco de dados, não podem gerar interrupções abruptas ou expor rastros técnicos de erro (*stack traces*) ao usuário final.

Como solução da arquitetura, adotou-se uma classe anotada com @ControllerAdvice. Esse recurso intercepta de maneira automatizada qualquer exceção originada nas camadas de serviço ou persistência, convertendo-a em respostas HTTP padronizadas. Em razão disso, a aplicação entrega mensagens claras ao *front-end*. Assim, um valor inválido gera uma resposta 400 com a mensagem do campo, e não um erro 500 com *stack trace*. As respostas de erro seguem um formato único, com data e hora, código de status, tipo de erro, mensagem e caminho da requisição, e cobrem os códigos 400, 401, 404, 405, 409 e 415. Erros inesperados são registrados no *log* do servidor, e o cliente recebe apenas uma mensagem genérica, sem rastros técnicos.

3.4. Monitoramento e validação da integração da API

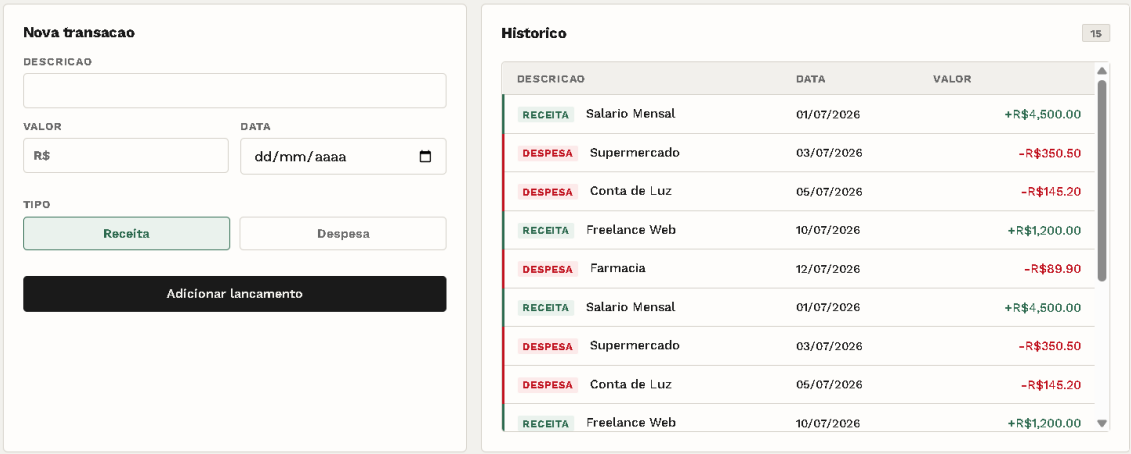
Para manter a comunicação entre as camadas da aplicação, foi estabelecida uma rotina de monitoramento e testes de integração. Este processo consistiu na validação dos *endpoints* REST, garantindo que as requisições enviadas pela interface Angular fossem processadas corretamente pela API Java e que as respostas retornadas mantivessem a integridade dos dados, focadas na precisão monetária. A validação foi realizada por meio de testes manuais com

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

a ferramenta Postman, em que requisições HTTP foram enviadas à API e os códigos de status de retorno foram verificados, como 201 para um lançamento válido, 400 para um dado incorreto e 401 para acesso sem autenticação, o que permitiu confirmar se o sistema comunicava ao cliente as condições de sucesso ou de violação das regras de negócio.

A precisão aritmética correta foi um pilar central na garantia da qualidade do *software*. Dado que o sistema lida com valores financeiros, foram realizados testes de soma e de cálculo de saldo pelo Postman, comparando os resultados da API com valores calculados manualmente. Essa comparação confirmou a exatidão dos cálculos realizados com BigDecimal. Com o tipo double, o saldo de uma receita de R$ 0,30 e uma despesa de R$ 0,10 resultava em 0,19999999999999998; com BigDecimal, em R$ 0,20, igual ao cálculo manual. A Figura 6 apresenta a interface de cadastro de movimentações e o histórico de transações.

Figura 6. Interface de cadastro de movimentações financeiras e histórico de transações



Fonte: Elaborado pelo autor (2026).

Por fim, o monitoramento dos *logs* de execução entre *Controller*, *Service* e *Repository* permitiu identificar e corrigir erros de mapeamento de objetos JSON.

3.5. Acessibilidade e experiência do usuário

Para muitos usuários, lidar com finanças constitui uma barreira, em razão da complexidade dos fatores citados anteriormente. Por isso, a interface foi desenvolvida em Angular, com organização simplificada, a fim de reduzir a dificuldade de uso.

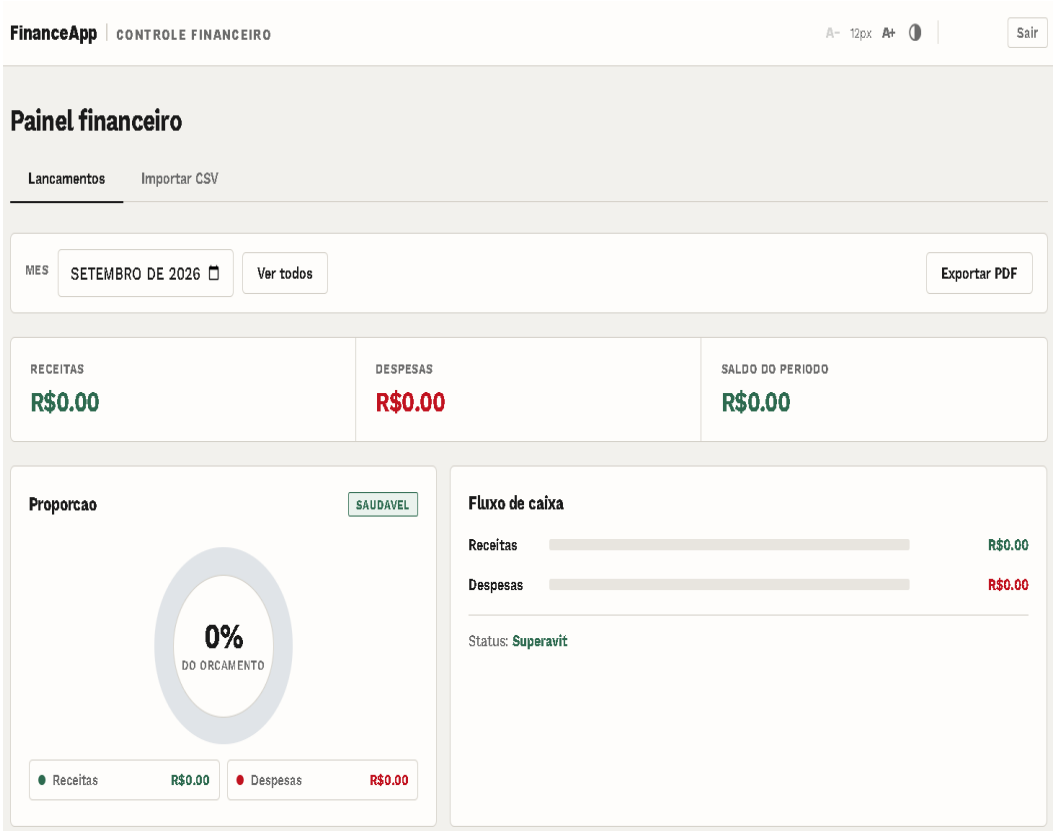
Ao criar as telas, foram escolhidos elementos visuais claros em paleta que atende ao contraste mínimo recomendado pela WCAG (W3C, 2024), nível AA. Essa escolha é relevante porque o contraste insuficiente de cores está entre as falhas de acessibilidade mais frequentes,

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

como observaram Batista e Baluz (2025) ao avaliar os sites de 20 universidades brasileiras. O sistema permite que o usuário acompanhe o saldo de suas finanças mês a mês.

A interface conta com recursos de personalização visual: botões para ajustar o tamanho da fonte (entre 12px e 24px) e alternar o contraste da interface, recursos visíveis na barra superior do painel apresentado na Figura 7. O modo de alto contraste troca a paleta por texto e bordas pretos sobre fundo branco (contraste de 21:1), com bordas mais espessas e destaque de foco visível, e a preferência fica salva no navegador. As cores verde e vermelha desse modo têm contraste de 10,0:1 e 8,8:1, acima do mínimo de 7:1 do nível AAA da WCAG (W3C, 2024), e os botões de acessibilidade têm rótulos para leitores de tela. Esses ajustes visam reduzir a barreira de uso para pessoas com dificuldades de visualização, sem exigir conhecimento técnico do usuário.

Figura 7. Painel de controle financeiro (*dashboard*) com barra de acessibilidade



Fonte: Elaborado pelo autor (2026).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.6. Arquitetura do sistema e padrão MVC

Para a estruturação do *software*, adotou-se o padrão arquitetural *Model-View-Controller* (MVC), fundamentado nos conceitos abordados no referencial teórico, com o propósito de assegurar a separação de responsabilidades e facilitar a manutenção do código ao longo do ciclo de vida da aplicação, permitindo uma evolução estruturada de todos os componentes envolvidos no projeto.

No âmbito do *Model*, concentram-se as entidades de negócio, a exemplo da classe Transacao, em conjunto com as rotinas de validação para preservar a integridade dos dados antes da persistência. A mediação entre as pontas do sistema recai sobre a camada de controle (*Controller*), responsável por gerenciar as requisições HTTP entre o *front-end* e o *back-end*, direcionando os fluxos para os serviços correspondentes (*Service Layer*).

Por sua vez, a camada *View* é representada pela interface desenvolvida em Angular, executada como cliente independente do *back-end*. Essa organização estrutural facilita a depuração e os testes, já que modificações na interface gráfica ocorrem de forma desacoplada da lógica de persistência no SQL Server.

3.7. Infraestrutura e ambiente de execução

Para que a plataforma funcionasse de forma consistente, a infraestrutura foi baseada em tecnologia de conteinerização por meio do Docker. Segundo a documentação oficial, o Docker permite empacotar e executar uma aplicação em um ambiente relativamente isolado, chamado contêiner, e o Docker Compose permite trabalhar com aplicações compostas por vários contêineres (DOCKER, 2026). O uso de *Dockerfiles* e do arquivo docker-compose.yml permite a orquestração do ambiente, garantindo que as versões do Java, do SQL Server e das dependências fiquem parametrizadas. Essa abordagem tem como principal vantagem a possibilidade de reduzir inconsistências entre diferentes ambientes de execução, facilitando também o processo de *deploy*.

4. RESULTADOS E DISCUSSÃO

Esta seção apresenta a validação da plataforma de gestão financeira desenvolvida, com foco na arquitetura, na precisão aritmética alcançada e na verificação dos requisitos estabelecidos nas seções anteriores.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

4.1. Validação da persistência e integridade de dados

Para garantir a precisão dos cálculos financeiros, foi utilizado o SQL Server como sistema gerenciador de banco de dados. A classe Transacao foi mapeada para a tabela transacoes, com o campo de valor do tipo DECIMAL(15,2), o que permitiu manter, também na camada de persistência, a precisão do tipo BigDecimal já discutida anteriormente. A tabela usuarios armazena o e-mail (único) e apenas o *hash* BCrypt da senha; a chave estrangeira usuario_id vincula cada lançamento ao seu usuário, e as consultas da API filtram os lançamentos pelo usuário autenticado, de modo que cada usuário acessa somente os próprios lançamentos. A Tabela 1 resume os casos de teste executados na API e os resultados obtidos.

Tabela 1. Casos de teste da API

Caso de teste	Entrada	Resultado esperado	Resultado obtido
Lançamento válido	Receita de R$ 150,00 com todos os campos	201 (criado)	201
Valor negativo	Valor de R$ -50,00	400 (dado inválido)	400
Campo obrigatório vazio	Descrição em branco	400 (dado inválido)	400
Acesso sem *login*	Requisição sem *token* JWT	401 (não autorizado)	401
Cálculo de saldo	Receita R$ 0,30 e despesa R$ 0,10	Saldo R$ 0,20	R$ 0,20 com BigDecimal Com double: 0,19999999999999998
Importação de CSV	Arquivo com uma linha de valor “fff”	Linha recusada, demais gravadas	Linha recusada com número e motivo

Fonte: Elaborado pelo autor (2026).

Esses resultados dialogam com Medeiros e Gonçalves (2020), que identificaram falhas na lógica matemática de planilhas eletrônicas perceptíveis apenas por usuários com conhecimento matemático. Na plataforma, a representação decimal exata e as restrições aplicadas no banco de dados retiram do usuário a responsabilidade de verificar a correção dos cálculos, o que é relevante para o público sem domínio avançado de matemática descrito na introdução. Quanto à segurança, o armazenamento apenas do *hash* BCrypt da senha está de acordo com as diretrizes do NIST, que determinam o armazenamento de senhas com *salt*, *hash* e fator de custo (TEMOSHOK *et al.*, 2025).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

4.2. *Design* da interface e recursos de acessibilidade

A interface apresenta retornos visuais imediatos para cada ação de salvamento ou erro, buscando reduzir a carga cognitiva do usuário. A acessibilidade foi tratada como atributo do próprio design, por meio de funcionalidades como o ajuste do tamanho da fonte e a alternância de contraste, permitindo que o usuário adapte a interface às suas necessidades visuais. Nas planilhas utilizadas na comparação, o tamanho do texto depende do zoom ou da formatação de cada célula, e o alto contraste depende das configurações do sistema operacional; na plataforma, os dois ajustes estão disponíveis em botões na própria interface.

Os recursos implementados respondem a falhas recorrentes apontadas na literatura: Batista e Baluz (2025) identificaram o contraste insuficiente de cores entre as falhas mais frequentes em sites de universidades brasileiras, e Melo *et al.* (2024) propuseram melhorias de acessibilidade para aplicativos bancários. Como a avaliação desses recursos não envolveu pessoas com deficiência visual, os resultados indicam aderência às recomendações da WCAG (W3C, 2024) no projeto da interface, e não efetividade de uso.

4.3. Comparação com o método tradicional (planilhas)

A comparação seguiu o procedimento descrito na seção 2.5 e considerou a facilidade de uso, a eficiência na execução das tarefas principais, desde o cadastro de transações até a consulta de saldo, a segurança e os recursos de acessibilidade.

Em relação aos lançamentos de transações, a organização do *design*, dos botões e campos, e a seleção direta do tipo de movimentação (receita ou despesa) fizeram com que o processo fosse mais direto do que o método manual em planilhas, nas quais a dispersão dos dados e a busca por colunas específicas, quanto maior a planilha, costumam tomar mais tempo do usuário. Referente à acessibilidade, os recursos implementados de ajuste do tamanho da fonte e do contraste da página permitem que o usuário final adapte a interface de acordo com suas necessidades, visando à facilidade visual, aspecto que os métodos tradicionais atendem de forma limitada. Quanto à segurança, o sistema conta com autenticação e persistência dos dados no banco de dados, em comparação com a exposição dos dados em arquivos locais, comum no uso de planilhas.

Em relação a trabalhos semelhantes, como o aplicativo de Andrade *et al.* (2024), voltado ao controle de despesas de estudantes universitários, a plataforma desenvolvida diferencia-se pela importação de dados de planilhas e pelos recursos de acessibilidade. Ressalta-se, contudo, que uma ferramenta de controle financeiro não substitui a educação financeira, cuja associação com menor propensão ao endividamento foi observada por Silva, Bubeck e Rodrigues Júnior (2025). De maneira geral, na avaliação realizada, a plataforma reuniu em um único ambiente os

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

aspectos de segurança, precisão dos cálculos, praticidade nos lançamentos e acessibilidade, o que sugere vantagens em relação ao controle manual em planilhas, a serem confirmadas em avaliações com usuários.

5. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver uma plataforma *web* centralizada de gestão financeira pessoal como alternativa às planilhas eletrônicas. Os resultados indicam que o objetivo foi atingido: a adoção do tipo BigDecimal (mapeado como DECIMAL no banco de dados) eliminou os erros de arredondamento observados nos testes, a camada de validação impediu a persistência de dados inconsistentes, e a autenticação com JWT e BCrypt, os recursos de acessibilidade e a importação de CSV foram implementados e integrados em um único ambiente. Dessa forma, o presente trabalho cumpre sua finalidade acadêmica ao empregar conceitos práticos da Engenharia de Computação na solução de uma demanda real, entregando uma alternativa estruturada e acessível para o planejamento financeiro individual.

Como limitações, a avaliação da plataforma foi qualitativa e baseada na observação do autor, sem testes com usuários reais, o que restringe a generalização dos resultados. Além disso, os testes da API foram realizados manualmente, os recursos de acessibilidade não foram validados com pessoas com deficiência nem por auditoria formal de conformidade, a importação aceita apenas arquivos CSV, e não foram realizadas medições quantitativas de desempenho.

Quanto às contribuições teóricas, o trabalho integra, em um problema concreto, conceitos de tipagem numérica, propriedades ACID, arquitetura MVC, API REST e segurança de aplicações *web*, evidenciando o impacto da representação de valores monetários nos cálculos financeiros. Reforça também a acessibilidade como requisito de projeto, e não como recurso complementar. Como contribuições práticas, entrega-se uma ferramenta funcional que permite migrar dados de planilhas existentes, com ambiente reprodutível via Docker, que pode servir de base para outros sistemas financeiros.

Como pesquisas futuras, sugere-se: (i) avaliar a plataforma com usuários reais, incluindo pessoas com deficiência visual, usando métricas de usabilidade; (ii) auditar a acessibilidade conforme o eMAG e as WCAG; (iii) ampliar a importação para outros formatos, como XLSX; (iv) integrar módulos de Inteligência Artificial para análise de despesas e recomendações automatizadas; (v) implementar a exportação de relatórios com gráficos; e (vi) implementar testes automatizados da API, por exemplo com JUnit e MockMvc.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Foram utilizadas as ferramentas de inteligência artificial generativa Claude (Anthropic) e Gemini (Google) como apoio durante a elaboração deste trabalho, entre novembro de 2025 e outubro de 2026. O Gemini foi utilizado em sugestões de estrutura do trabalho e no levantamento inicial de referências bibliográficas, realizado a partir do final de 2025, em etapa anterior do projeto, e as referências foram posteriormente verificadas pelo primeiro autor nas fontes originais. O Claude foi utilizado na estruturação, na organização e na revisão textual do artigo e na tradução dos resumos para o inglês e o espanhol. Em todos os casos, os conteúdos gerados foram criticamente avaliados, modificados e validados pelos autores, que assumem responsabilidade integral pelos resultados e pelas referências apresentados, incluindo eventuais erros ou imprecisões. O texto do artigo foi lido e revisado pelos três autores.

FINANCIAMENTO

Esta pesquisa não recebeu financiamento de agências de fomento, instituições ou empresas.

CONFLITO DE INTERESSES

Os autores declaram não haver conflitos de interesses.

REFERÊNCIAS

ANDRADE, I. K. M. *et al.* Aplicativo para controle financeiro de estudantes universitários. **Revista de Gestão e Secretariado**, v. 15, n. 4, p. 1-20, 2024. DOI: https://doi.org/10.7769/gesec.v15i4.3641.

BATISTA, H. E. N.; BALUZ, R. A. R. S. Avaliação de sites das instituições de ensino superior de acordo com as diretrizes de acessibilidade para conteúdo web (WCAG 2.1): uma análise comparativa das principais universidades federais e estaduais do Brasil. **iSys: Revista Brasileira de Sistemas de Informação**, v. 18, n. 1, p. 14:1-14:26, 2025. DOI: https://doi.org/10.5753/isys.2025.5401.

BRASIL. Governo Digital. **Do eletrônico ao digital**: linha do tempo. Brasília, 25 nov. 2019. Disponível em: https://www.gov.br/governodigital/pt-br/estrategia-de-governanca-digital/do-eletronico-ao-digital. Acesso em: 19 out. 2026.

BRASIL. Ministério do Planejamento, Orçamento e Gestão. **eMAG**: Modelo de Acessibilidade em Governo Eletrônico. Versão 3.1. Brasília: MP, 2014. Disponível em: https://emag.governoeletronico.gov.br/. Acesso em: 22 mar. 2026.

DEITEL, P.; DEITEL, H. **Java**: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

DOCKER. **Docker overview**. [S. l.]: Docker Inc., 2026. Disponível em: https://docs.docker.com/get-started/docker-overview/. Acesso em: 18 out. 2026.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson, 2019.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. RFC 7519. [S. l.]: IETF, 2015. Disponível em: https://www.rfc-editor.org/rfc/rfc7519. Acesso em: 10 out. 2026.

KROENKE, D. M.; AUER, D. J.; VANDENBERG, S. L.; YODER, R. C. **Database processing**: fundamentals, design, and implementation. 15. ed. Boston: Pearson, 2018.

MDN WEB DOCS. **SPA (Single-page application)**. [S. l.]: Mozilla, 11 jul. 2025. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/SPA. Acesso em: 10 out. 2026.

MEDEIROS, J. de; GONÇALVES, R. A. Planilhas eletrônicas de cálculo: inconsistências, erros e divergências. *In:* MARTINS, E. R. (org.). **Ciência da computação e tecnologias digitais**: contribuições na solução de problemas. Curitiba: Bagai, 2020. cap. 6, p. 72-84. Disponível em: https://educapes.capes.gov.br/handle/capes/585221. Acesso em: 15 mar. 2026.

MELO, G. L. N. *et al.* Inspecting the accessibility of chatbots and mobile banking for emergent users in the context of people with low literacy. **Journal on Interactive Systems**, v. 15, n. 1, p. 839-859, 2024. DOI: https://doi.org/10.5753/jis.2024.4317.

MORAES, G.; MISAEL, V.; BRAYNER, A. From legacy to modern systems: an enterprise-ready workflow for database migrations. **Journal of Information and Data Management**, v. 17, n. 1, p. 210-218, 2026. DOI: https://doi.org/10.5753/jidm.2026.5730.

SERASA EXPERIAN. **Mapa da inadimplência e negociação de dívidas no Brasil**. São Paulo: Serasa Experian, jan. 2026. Disponível em: https://www.serasa.com.br/limpa-nome-online/blog/mapa-da-inadimplencia-e-renogociacao-de-dividas-no-brasil/. Acesso em: 15 mar. 2026.

SILVA, J. M. N. e; BUBECK, S. K.; RODRIGUES JÚNIOR, M. M. Efeito da educação financeira e atitudes frente ao dinheiro na propensão ao endividamento. **Revista Mineira de Contabilidade**, v. 26, n. 2, p. 10-23, 2025. DOI: https://doi.org/10.51320/rmc.v26i2.1615.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SPRING. **Spring Boot**. [S. l.]: Broadcom, c2026. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 10 out. 2026.

TEMOSHOK, D. *et al.* **Digital identity guidelines**: authentication and authenticator management. Gaithersburg: National Institute of Standards and Technology, 2025. (NIST Special Publication 800-63B-4). DOI: https://doi.org/10.6028/NIST.SP.800-63b-4.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.2**. [S. l.]: W3C, 12 dez. 2024. Disponível em: https://www.w3.org/TR/WCAG22/. Acesso em: 15 out. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.